Final Narrative

Over the last 8 weeks I have taken four separate files, all independent of each other, and consolidated them into one functioning program. The original artifacts consisted of a binary search tree, that when run, created a bid explorer program. There was also a linked list, that also created a bid explorer program. Finally, there was a hash table, that again, created a bid explorer program. All three files accomplished the same goal but utilized different data structures. What I chose to do was combine all of these files into a single program that allowed the user to choose which data structure to use, and then from there the user could add, find, display or remove bids.

The first two narratives highlight the changes and enhancements for the Data Structures and Algorithms, and the Database Categories. This final narrative will highlight the overall accomplished enhancements and how it showcases my strengths. The enhancements are as follows, for Software Design and Engineering, I created one solid program and developed the code necessary to allow the user to select which data structure they wanted to use. The code does conform to current coding standards, such as declaring global variables at the top of the file. I also broke the code down into helper functions, commented on most of the functions, and used multiple classes to break up the architecture. The code is consistently formatted, and mostly well structured and consistent in style. There are a couple redundant functions that were duplicated a few times, such as strToDouble. There is no unreachable code, however there are one or two empty functions for loading data structures. There are some comments that were left in for debugging purposes. There are one or two functions that could be replaced by a library function or utility class i.e., the string to double method. There is one block, in BidProgram, for loading the linked list which occurs twice, but the helper function is only used once. The was done to satisfy the compiler. By having thorough and clear comments, proper formatting, and the files consolidated to one program I have created an environment that supports collaboration. These enhancements allow other developers that follow me to know what my code is, what it does, and allow them the opportunity to collaborate, or enhance it themselves. I ensured that my code, comments, and formatting, were professional and coherent. I ensured that my code was as technically sound as possible and adapted to ensure functionality for the user. I found storage use is mostly efficient. One inefficiency is that the binary search tree is built on bidIds that are linearly increasing. So, the tree resembles a linked list rather than a tree. A very small number of “magic number” and string constants are used. A utility module was planned to be built, however due to reasons of compilation this didn’t happen. Other than that, modules are well architected and not overly complex.

The best example of an enhancement for the data structures and algorithms category is that the code avoids comparing floating-point numbers for equality and mostly prevents rounding errors. Some of the calculations for total bidValue and meanBid value have fractional cents values, meaning values less than a penny. The code mostly avoids addition and subtractions on numbers with greatly different magnitudes, however, when totaling bid value we were adding bid values of less than $1000 to the total amount, which is over 5 million. Divisors are mostly tested for zero and/or noise. Additionally, all loops, branches and logic constructs are complete, correct and properly nested. The IF-ELSE chains are written to be logical, meaning that reasonable test cases come first, such as empty test cases. These are not always the most common test cases that occur first. I did ensure that all cases in an IF-ELSEIF chain block include ELSE, or DEFAULT, clauses. Additionally, every case statement does contain a default. Loop termination conditions are not always obvious, but they are achievable. Indexes and subscripts ARE properly initialized prior to the loops. Technically, some of the statements could be placed outside the loops. They should not be though, to keep consistent with proper coding practices. The code mostly avoids manipulating the index variable. Sometimes the indexes, pointers, and subscripts are tested against array, record, or file bounds. The imported data was checked a fair amount however, improper formatting, of the imported data, was discovered and corrected as much as possible. All output variables are assigned, and the correct data is operated on in each statement. Sometimes, auction value is operated on as a string when it is an integer. But, when necessary, it is converted to an int. One area for improvement is that memory allocations are not always deallocated. There is no external device access that occurs in the code. Files are checked for existence before attempting to access them. Also, when the program is terminated, all files and devices are left in the correct state. Finally, I had to alter the way the data was calculated so my numbers were as accurate as possible. When you have an app that stores bid data, the last thing that is wanted is mathematical or algorithm discrepancies. At one point during this process I had an issue where my final bids were being rounded. This alters all the corresponding data and would invalidated the data mining feature of the software, had I not resolved the problem.

By adding data mining to the program, I met the requirements for the database category while adding a whole new layer to my program that expands upon its usefulness exponentially. Giving the user the ability to identify patterns and common items bid on, allows them to tailor the items the post to maximize their profits, all while giving the bidders access to the items they want. With the data set that I had there were a total of 900 bids. When running the data miner it notifies the user that with this current dataset the minimum bid is $1, and the maximum bid is $17,600. The dataminer also shows the minimum, and maximum bid Ids. Additionally, the software shows the minimum, and maximum auction fees, as well as the total amount of auctions fees. This is essential in letting the user know what the income looks like with the data they present.

By utilizing various platforms for coding, I showcased my abilities as a programmer, my skills, and my understanding of the software engineering process. Visual Studio 2022 was essential in allowing me access to an IDE that works well for the type of program I was creating. By ensuring that my code was properly tested and written with proper formatting I maintained a security mindset. With no client’s personal data being stored in the database, there was less of a need to be overly vigilant.

Overall, my experience over the last 8 weeks has been fulfilling. I finally felt like I understood what I was doing, how to accomplish set tasks, and showcase my ability in meeting the client’s needs. I have struggled with GitHub some and trying to get everything uploaded and functioning properly, within my ePortfolio.